PATENT MS158541.01/MSFTP192US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:			
Applicant(s):	Brian K. Pepin, *et al.*	Examiner:	Andre R. Fowlkes
Serial No:	09/812,207	Art Unit:	2192
Filing Date:	March 19, 2001		

Title: SYSTEM AND METHOD TO FACILITATE DESIGN-TIME COMPONENT DISCOVERY

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

Appellants submit this brief in connection with an appeal of the above-identified patent application. A credit card payment form is filed concurrently herewith in connection with all fees due regarding this appeal brief. In the event any additional fees may be due and/or are not covered by the credit card, the Commissioner is authorized to charge such fees to Deposit Account No. 50-1063 [MSFTP192US].

I. Real Party in Interest (37 C.F.R. §41.37(c)(1)(i))

The real party in interest in the present appeal is Microsoft Corporation, the assignee of the present application.

II. Related Appeals and Interferences (37 C.F.R. §41.37(c)(1)(ii))

Appellants, appellants' legal representative, and/or the assignee of the present application are not aware of any appeals or interferences which may be related to, will directly affect, or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. Status of Claims (37 C.F.R. §41.37(c)(1)(iii))

Claims 1-22 stand rejected by the Examiner. The rejection of claims 1-22 is being appealed.

IV. Status of Amendments (37 C.F.R. §41.37(c)(1)(iv))

No claims were amended subsequent to final rejection.

V. Summary of Claimed Subject Matter (37 C.F.R. §41.37(c)(1)(v))

A. Independent Claim 1

Independent claim 1 recites an application development system, comprising: a development tool that facilitates application development (*see, e.g.*, page 7, ll. 15-26) in a design time environment (*see, e.g.*, page 8, line 29-page 9, line 7) and reports at least one of simulated run time and compile time information based upon design time attributes (*see, e.g.*, page 10, line 17-page 11, line 4); a software component (*see, e.g.*, FIG. 1, item 150); and a type descriptor that accesses metadata associated with the software component, the type descriptor dynamically provides information associated with an instance of the software component to the development tool (*see, e.g.*, page 7, ll. 15-26).

B. Independent Claim 8

Independent claim 8 recites an application development system, comprising: a development tool that facilitates application development in a design time environment and

reports at least one of simulated run time and compile time information based upon design time attributes; and a software component that provides a custom type descriptor interface which provides information associated with the software component to a type descriptor (*see, e.g.*, page 8, ll. 8-26; FIG. 4), the type descriptor accesses metadata associated with the software component and the type descriptor receives manipulated information associated with the software component from the custom type descriptor interface and provides one of the metadata or information received from the custom type descriptor interface to the development tool. (*See, e.g.*, page 7, ll. 15-26; page 8, line 29-page 9, line 7; page 10, line 17-page 11, line 4; FIG. 1).

C. **Independent Claim 14**

Independent claim 14 recites an application development system, comprising: a plurality of contained components (*see, e.g.*, page 10, ll. 4-16); a development tool that reports at least one of simulated run time and compile time information based upon design time attributes to facilitate application development and that provides a container that facilitates communication among the plurality of contained components and further provides a site comprising a plurality of type descriptor filter service interfaces for manipulating information associated with the plurality of contained components; and a type descriptor that accesses metadata associated with the plurality of contained components the type descriptor receives information associated with a design time behavior of the plurality of contained components from the plurality of type descriptor filter service interfaces and provides one of the metadata and information received from the plurality of type descriptor filter service interfaces to the development tool. (*See, e.g.*, page 7, ll. 15-26; page 8, line 29-page 9, line 7; page 10, line 17-page 11, line 4; page 11, line 14-page 12, line 7; FIGS. 1, 5).

D. **Independent Claim 16**

Independent claim 16 recites in a component based environment, a method for developing an application comprising: determining whether an instance of a component implements a custom type interface; invoking the custom type descriptor interface of the instance of the component, the custom type descriptor interface manipulating compile time information regarding the instance of the component for simulating component behavior at one of a design time and a runtime; receiving the manipulated information regarding the instance of the

component from the custom type descriptor interface; and storing the manipulated information regarding the instance of the component. (*See, e.g.*, page 12, line. 25-page 14, line 9; FIGS. 6, 7)

E. Independent Claim 18

Independent claim 18 recites in a component based environment, a method for developing an application comprising: receiving information regarding an instance of a component; determining whether the instance of the component is contained by a container; determining whether any other contained component desires to modify information regarding the instance of the component; modifying the information regarding the instance of the component; determining whether the container implements a type descriptor filter service interface for the instance of the component; manipulating the compile time information regarding the instance of the component by the type descriptor filter service interface for simulating component behavior at one of a design time and a runtime; and storing the manipulated information regarding the instance of the component. (*See, e.g.*, page 10, line. 4-page 14, line 9; FIGS. 5-7)

F. Independent Claim 21

Independent claim 21 recites an application development system comprising: means for determining whether an instance of a component implements a custom type descriptor interface (*see, e.g.*, page 9, ll. 8-26; page 10, line 25-page 11, line 3; page 12, line 25-page 13, line 3; page 13, line 17-page 14, line 2; FIGS. 4-7); means for invoking the custom type descriptor interface of the instance of the component, the custom type descriptor interface manipulates information regarding the instance of the component (*see, e.g.*, page 9, ll. 8-26; page 10, line 25-page 11, line 3; page 12, line 25-page 13, line 3; page 13, line 17-page 14, line 2; FIGS. 4-7); means for manipulating compile time information regarding the instance of the component for simulating component behavior at one of a design time and a runtime; (*see, e.g.*, page 9, line 27-page 10, line 3; FIG. 4); means for receiving the manipulated information regarding the instance of the component from the custom type descriptor interface (*see, e.g.*, page 9, ll. 8-26; page 10, line 25-page 11, line 3; page 12, line 25-page 13, line 3; page 13, line 17-page 14, line 2; FIGS. 4-7); and means for storing the manipulated information regarding the instance of the component (*see, e.g.*, page 8, ll. 2-4; page 8, ll. 18-28; page 9, ll. 10-26; page 11, ll. 5-13; page 12, ll. 4-7; FIGS. 1-5).

The aforementioned means for limitations are identified as claim elements subject to the provisions of 35 U.S.C. §112 ¶6. The corresponding structures are identified with reference to the specification and drawings in the parentheticals above corresponding to those claim limitations.

VI. Grounds of Rejection to be Reviewed (37 C.F.R. §41.37(c)(1)(vi))

A. Whether claims 1-22 are unpatentable under 35 U.S.C. §102(b) over Sarkar (US 6,012,067).

VII. Argument (37 C.F.R. §41.37(c)(1)(vii))

A. Rejection of Claims 1-22 Under 35 U.S.C. §102(b)

Claims 1-22 stand rejected under 35 U.S.C. §102(b) as being anticipated by Sarkar (US 6,012,067). Reversal of this rejection is respectfully requested for at least the following reasons. Sarkar does not teach or suggest each and every feature recited in the subject claims.

> "A claim is anticipated only if ***each and every element*** as set forth in the claim is found, either expressly or inherently described in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ 2d 1051, 1053 (Fed. Cir. 1987). Emphasis added. "***The identical invention must be shown in as complete detail as is contained in the...claim.***" Richardson v. Suzuki Motor Co., 868 F.2d 1226, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

Independent claims 1, 8, and 14

The claimed subject matter relates to application development in a design time environment and/or discovering the design time attributes of a component and simulating how a component behaves at run time without the need to reconstruct *(e.g.*, recompile) the component. In particular, independent claim 1 (and similarly independent claims 8 and 14) recites, "a development tool that facilitates ***application development*** in a ***design time environment*** and ***reports at least one of simulated run time and compile time information based upon design time attributes*** and ***a type descriptor*** that ... provides information associated with an instance of

the software component to ***the development tool.***" Sarkar does not disclose or suggest these novel features.

Rather, Sarkar relates to a multi-tier client/server architecture, wherein the first tier is a web browser on the client side (*see* Fig. 1; col. 6, ll. 55-56), with business logic (second tier) and a relational database (third tier) on the server side (*see* Fig. 1; col. 7, ll. 1-3). Sarkar puts the business logic inside the relational database, thereby eliminating the middle tier. (See col. 7, ll. 5-9; col. 10, ll. 30-45). Sarkar employs the native abilities of URLs that can access web objects, as well as the native abilities of relational database server links for remote distributed databases and object pointers to reference any kind of object within the relational database (which now includes business logic). (*See* col. 5, ll. 19-35). Thus, a mechanism is employed *to manipulate objects in relational databases over the internet.* (*See* col. 5, ll. 11-14). This mechanism is an SQL query or an ORB request carrying the SQL query (*see* Abstract; col. 5, ll. 58-65; col. 7, ll. 3-5; col. 10, ll. 35-38) that can address (*e.g.*, represent) objects over the internet *via* URL's, and the SQL query manipulates objects when the SQL query is *executed*, which occurs at run-time. (*See* col. 5, ll. 65-68; col. 7, ll. 3-5). However, Sarkar is devoid of any teaching of manipulating objects in ***a design time environment.*** Further, the types of objects Sarkar can manipulate include web objects and business objects. Web objects include text, images, audio, video, and HTML pages. (*See* col. 5, line 17; col. 5, ll. 21-22; col. 2, ll. 34-35; col. 10, ll. 54-55). Business object include results of queries across distributed relational database with business application logic applied to those results, CORBA objects and other objects within a relational database such as tables. (*See* col. 5, ll. 14-17; col. 2, ll. 32-33; col. 5, ll. 25-26).

Thus, Sarkar provides a mechanism (*i.e.*, SQL query) for representing (*i.e.*, access *via* a URL) and manipulating objects (*i.e.* web objects or relational data) in relational databases over the Internet in a run-time environment (*e.g.*, as the SQL query is executed). At page 3 of the Advisory Action (dated March 10, 2006), the Examiner broadly interprets the phrase recited in Sarkar "a mechanism for representing and manipulating objects in relational databases" as anticipatory of the subject claims which recite, "***a development tool that facilitates application development in a design time environment.***" The Examiner substitutes several words of the aforementioned phrase in Sarkar for words in the claims, and incorrectly concludes that the claims are anticipated by virtue of giving the claims "their broadest reasonable interpretation consistent with the specification." *In re Hyatt*, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667

(Fed. Cir. 2000). However, the Examiner is not attempting to give a reasonable interpretation of the *pending claims*, but rather, suggests a different scope of the words in a single phrase *of the cited art*, and, moreover, the scope suggested by the Examiner for the phrase in Sarkar is not in any manner consistent with the teachings of Sarkar. For example, the Examine suggests that "a mechanism" can be read as "a development tool" ostensibly because the Examiner reasons that "a development tool" could also be named "a mechanism," irrespective of the fact that what is described as "a mechanism" in Sarkar is not remotely similar to "a development tool" of the subject claims. The Examiner is employing a single phrase and very little else from Sarkar, taken dramatically out of the context and scope of the reference, in order to reject the subject claims. To that end, the Examiner suggests that "a mechanism" is "a development tool"; that "objects" are "applications"; that "manipulating objects" is "application development"; and that "in relational databases" is equivalent to "a design time environment."

a. Sarkar does not teach or suggest a development tool that facilitates application development in a design time environment.

In particular, as indicated in the column and line references above, Sarkar teaches that the mechanism described is an SQL query (that is able to address objects *via* URL's), and that this SQL query can manipulate objects in relational databases. An SQL query is not in any sense disclosed in Sarkar or known in the art as "***a development tool that facilitates application development*** in a design time environment." Similarly, objects in a relational database (*e.g.*, business objects such as relational data tables, records, and results; or web objects such as text, images, audio, and video) are not disclosed by Sarkar or known in the art to be applications, but rather are objects. In fact, Sarkar expressly discloses that there is no distinction between an *application* and a *server* (*see* col. 5, line 67 – col. 6, line 1; col. 10, ll. 33-35), but does distinguish between *servers* and *objects*. Hence, the Examiner's interpretation that objects are applications is summarily contradicted by the reference (*i.e.*, an application is equivalent to a server, but an object is not equivalent to a server, therefore, an application is not equivalent to an object). Therefore, accessing and/or manipulating, *e.g.*, text or images in a relational database, as Sarkar teaches and the Examiner incorrectly indicates as anticipatory (*see* col. 5, ll. 49-56), is materially distinct from ***a development tool that facilitates application development*** in a design time environment recited in the subject claims. Accordingly, this rejection should be reversed.

*b. **Sarkar does not teach or suggest a development tool that facilitates application development in a design time environment.***

Likewise, manipulating objects in a relational database is not disclosed by Sarkar or known in the art to be identical to ***application development in a design time environment.*** That is, manipulating, *e.g.*, data tables is not application development, relational databases are not design time environments for application development, and, moreover, the "manipulating" as taught by Sarkar is a result of an SQL query that occurs as the query *executes*, which teaches manipulation of objects during *run-time*, not *design time*. Accordingly, despite the Examiner's assertions to the contrary, a mechanism (*i.e.*, SQL query) for representing (*i.e.*, access *via* a URL) and manipulating objects (*i.e.* web objects or relational data) in relational databases does not teach or suggest a development tool that facilitates ***application development in a design time environment***. Rather, Sarkar teaches, *e.g.*, joining relational database tables as the query *executes* (*i.e.*, during run time). Thus, this rejection should be reversed.

*c. **Sarkar does not teach or suggest a development tool that...reports at least one of simulated run time and compile time information based upon design time attributes.***

Furthermore, Sarkar does not teach or suggest "a development tool that...reports at least one of ***simulated run time*** and ***compile time*** information based upon design time attributes." The reference is silent with respect to reporting simulated run time and compile time information, let alone such information being based upon design time attributes of which Sarkar is also silent. The Examiner attempts to circumvent these limitations by suggesting that the word "representing" disclosed in Sarkar can be substituted by "simulating." Such a substitution is inappropriate on its face because Sarkar discloses that objects can be accessed (*i.e.*, represented) over the Internet *via* URL's. Allowing an object to be accessed (represented) over the Internet is not simulating, and particularly not ***simulated run time***. Therefore, the Examiner's substitution is not within the scope or context of the teachings of the reference and such an argument is inappropriate However, even if it were appropriate to substitute "simulating" for "representing," simulating an object in a relational database is materially distinct from reporting ***simulated run time*** ... information ***based upon design time attributes***. Sarkar is completely void of teachings related to simulated run time and compile time information based upon design

time attributes. The Examiner does not identify what design time attributes Sarkar possesses, much less how such could be used to report either simulated run time and/or compile time information. Rather, the Examiner broadly suggests that some form of simulating occurs in Sarkar (based solely on the fact that the term "representing" is disclosed), and, as such, that the claims are anticipated without regard to reasonable analysis of the claim as a whole. Accordingly, it is respectfully requested that this rejection be reversed.

d. Sarkar does not teach or suggest a type descriptor that ... provides information ... to the development tool.

Moreover, at page 3 of the Final Office Action (dated November 2, 2005), the Examiner argues that the SQL query (*e.g.*, the "mechanism") is the development tool, yet at page 4 the Examiner argues that the SQL query is the type descriptor, whereas the subject claims recite "*a type descriptor* that ... provides information ... to *the development tool*." It is readily apparent that an SQL query is not capable of teaching or suggesting either a type descriptor or the development tool of the subject claims, let alone stand for both claimed aspects simultaneously. Sarkar does not teach or suggest an SQL query that provides information to the SQL query, and such an implication is erroneous on its face. In essence, the Examiner is employing a single feature taught in Sarkar to represent multiple, distinct features of the instant claims and simply disregards the fact that the claims recite one feature (*e.g.*, a type descriptor) that provides information to the other feature (*e.g.*, the development tool).

For at least the reasons detailed herein, Sarkar does not teach the identical invention in as complete detail as the subject claims. In accordance therewith, this rejection of independent claims 1, 8, 14 as well as all claims that depend there from, should be reversed.

Independent claims 16, 18, and 21

The claimed subject matter further relates to a component based environment wherein static metadata can be compiled in a component and can then be dynamically adjusted to reflect modifications in design time. In accordance therewith, run time behavior of a component can be simulated based upon design time manipulations without recompiling the component. In particular, independent claim 16 (and similarly independent claims 18 and 21) recites, "manipulating ***compile time information regarding the instance of the component...for***

simulating component behavior at one of a design time and a runtime". Sarkar does not disclose or suggest these novel features.

At page 8, the Final Office Action contends Sarkar discloses the aforementioned features at col. 5, ll. 11-14. However, the indicated portions simply recite, "*It is a primary objective of the present invention to provide a mechanism for representing and manipulating heterogeneous objects in relational databases over the internet.*" As with the arguments provided in connection with the rejection of independent claims 1, 8 and 14, the Examiner is again incorrectly construing a mechanism for representing objects over the Internet as equivalent to ***simulating component behavior at one of a design time and a runtime***. Furthermore, Sarkar is void of any teaching or suggestion of manipulating ***compile time information regarding the instance of the component...for simulating component behavior at one of a design time and a runtime***. Moreover, the Examiner does not argue to the contrary. Rather, the extent to the Examiner's analysis is to argue the term "representing" is tantamount to "simulating" and further analysis, *e.g.*, analysis of the claim as a whole, is not necessary. Accordingly, reversal of this rejection of independent claims 16, 18, and 21, as well as all associated dependent claims, is requested.

B. Conclusion

For at least the above reasons, the claims currently under consideration are believed to be patentable over the cited references. Accordingly, it is respectfully requested that the rejections of claims 1-22 be reversed.

If any additional fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063 [MSFTP192US].

Respectfully submitted,
AMIN, TUROCY & CALVIN, LLP

Patrica S. Murphy
Reg. No. 55,964

AMIN, TUROCY & CALVIN, LLP
24th Floor, National City Center
1900 East 9th Street
Telephone: (216) 696-8730
Facsimile: (216) 696-8731

VIII. Claims Appendix (37 C.F.R. §41.37(c)(1)(viii))

1. An application development system, comprising:

a development tool that facilitates application development in a design time environment and reports at least one of simulated run time and compile time information based upon design time attributes;

a software component; and

a type descriptor that accesses metadata associated with the software component, the type descriptor dynamically provides information associated with an instance of the software component to the development tool.

2. The system of claim 1, the type descriptor stores the information associated with the software component.

3. The system of claim 1, the information associated with the software component comprises at least one of types, members, attributes, properties and events.

4. The system of claim 1, the software component provides a custom type descriptor interface.

5. The system of claim 4, the custom type descriptor interface manipulates information associated with the software component and provides one of design time information and runtime information to the type descriptor.

6. The system of claim 5, the information provided to the development tool is one of the metadata and the manipulated information received from the custom type descriptor interface.

7. The system claim 1, the type descriptor provides the information associated with the software component in response to a request from a developer.

8. An application development system, comprising:

a development tool that facilitates application development in a design time environment and reports at least one of simulated run time and compile time information based upon design time attributes; and,

a software component that provides a custom type descriptor interface which provides information associated with the software component to a type descriptor, the type descriptor accesses metadata associated with the software component and the type descriptor receives manipulated information associated with the software component from the custom type descriptor interface and provides one of the metadata or information received from the custom type descriptor interface to the development tool.

9. The system of claim 8, the custom type descriptor interface manipulates the information associated with the software component to determine one of a design time and a runtime behavior of the software component before providing the information to the type descriptor.

10. The system of claim 8, the type descriptor stores one of the metadata or information received from the custom type descriptor interface.

11. The system of claim 8, the metadata comprises at least one of types, members, attributes, properties and events.

12. The system of claim 8, the information provided by the custom type descriptor interface comprises at least one of types, members, attributes, properties and events.

13. The system claim 8, where the type descriptor provides the information associated with the software component in response to a request from a developer.

14. An application development system, comprising:

a plurality of contained components;

a development tool that reports at least one of simulated run time and compile time information based upon design time attributes to facilitate application development and that provides a container that facilitates communication among the plurality of contained components and further provides a site comprising a plurality of type descriptor filter service interfaces for manipulating information associated with the plurality of contained components; and

a type descriptor that accesses metadata associated with the plurality of contained components the type descriptor receives information associated with a design time behavior of the plurality of contained components from the plurality of type descriptor filter service interfaces and provides one of the metadata and information received from the plurality of type descriptor filter service interfaces to the development tool.

15. The system of claim 14 at least one of the contained components provides a custom type descriptor interface.

16. In a component based environment, a method for developing an application comprising:

determining whether an instance of a component implements a custom type interface;

invoking the custom type descriptor interface of the instance of the component, the custom type descriptor interface manipulating compile time information regarding the instance of the component for simulating component behavior at one of a design time and a runtime;

receiving the manipulated information regarding the instance of the component from the custom type descriptor interface; and,

storing the manipulated information regarding the instance of the component.

17. The method of claim 16, further comprising at least one of the following acts:

receiving a request from a development tool for information regarding the instance of the component;

discovering metadata associated with the instance of the component; and,

reporting the information regarding the instance of the component to the development tool.

18. In a component based environment, a method for developing an application comprising:

receiving information regarding an instance of a component;

determining whether the instance of the component is contained by a container;

determining whether any other contained component desires to modify information regarding the instance of the component;

modifying the information regarding the instance of the component;

determining whether the container implements a type descriptor filter service interface for the instance of the component;

manipulating the compile time information regarding the instance of the component by the type descriptor filter service interface for simulating component behavior at one of a design time and a runtime; and,

storing the manipulated information regarding the instance of the component.

19. The method of claim 18, further comprising at least one of the following acts:

receiving a request from a development tool for information regarding the instance of the component;

discovering metadata associated with the instance of the component;

determining whether the instance of the component implements a custom type interface;

invoking the custom type descriptor interface of instance of the component, the custom type descriptor interface manipulating information regarding the instance of the component;

manipulating information regarding the instance of the component;

receiving information regarding the instance of the component from the custom type descriptor interface; and,

reporting the information regarding the instance of the component to the development tool.

20. A computer-readable medium having computer-executable instructions for executing at least a portion of the method of claim 16.

21. An application development system comprising:

means for determining whether an instance of a component implements a custom type descriptor interface;

means for invoking the custom type descriptor interface of the instance of the component, the custom type descriptor interface manipulates information regarding the instance of the component;

means for manipulating compile time information regarding the instance of the component for simulating component behavior at one of a design time and a runtime;

means for receiving the manipulated information regarding the instance of the component from the custom type descriptor interface; and,

means for storing the manipulated information regarding the instance of the component.

22. The application development system of claim 21, further comprising at least one of means for receiving a request from a development tool for information regarding the instance of the component, means for discovering metadata associated with the instance of the component and means for reporting the information regarding the instance of the component to the development tool.

IX. Evidence Appendix (37 C.F.R. §41.37(c)(1)(ix))

None.

X. Related Proceedings Appendix (37 C.F.R. §41.37(c)(1)(x))

None.